GL2PS: an OpenGL to PostScript printing library

Christophe Geuzaine

Version 0.72, 21 January 2003

Contents

1 Introduction

GL2PS is a library for creating high quality vector output (primarily PostScript) from any OpenGL application. The main difference between GL2PS and other similar libraries (see section 6) is the use of sorting algorithms capable of handling intersecting and stretched polygons, as well as non manifold objects.

The library, written in C, is released under the GNU Library General Public License (see `http://www.gnu.org/` for more details), and is available at `http://www.geuz.org/gl2ps/`. Any corrections, questions or suggestions should be e-mailed to `geuz@geuz.org`.

The interface consists of eight functions, all beginning with the prefix `gl2ps`. All the data structures and the symbolic constants peculiar to GL2PS begin with `GL2PS`.

2 Usage

2.1 `gl2psBeginPage` and `gl2psEndPage`

2.1.1 Specification

```
void gl2psBeginPage( const char *title, const char *producer,
                     GLint format, GLint sort, GLint options,
                     GLint colormode, GLint colorsize,
                     GL2PSrgba *colortable, GLint buffersize,
                     FILE *stream, const char *filename )

void gl2psEndPage( void )
```

2.1.2 Description

`gl2psBeginPage` and `gl2psEndPage` delimit the OpenGL commands that will be caught in the feedback buffer and output to `stream`. The parameters given to `gl2psBeginPage` determine the way primitives are handled:

`title` Specifies the plot title. For PostScript output, this string is placed in the `%%Title` field.

`producer` Specifies the plot producer. For PostScript output, this string is placed in the `%%For` field.

`format` Specifies the output format, chosen among:

- `GL2PS_PS` The output stream will be a PostScript file.
- `GL2PS_EPS` The output stream will be an Encapsulated PostScript file.
- `GL2PS_TEX` The output stream will be a LaTeX file, containing only the text strings of the plot (cf. section 2.2), as well as an `\includegraphics` command permitting to include a graphic file having the same basename as `filename`.[1]

[1] The two steps to generate a LaTeX plot with GL2PS are thus:

1. generate the PostScript file (e.g. `file.ps`) with no text strings, using the `GL2PS_PS` or `GL2PS_EPS` format combined with the `GL2PS_NO_TEXT` option;
2. generate the LaTeX file `file.tex`, using the `GL2PS_TEX` format and specifying `file.tex` as the `filename` argument to `gl2psBeginPage`.

You can of course combine the LaTeX output with other graphic formats than PostScript. For example, you may transform `file.ps` into `file.pdf` and use pdfLaTeX with the same `file.tex` as for PostScript. You may also use a bitmap image (e.g. Jpeg or png) and still combine it with `file.tex`.

`sort` Specifies the sorting algorithm, chosen among:

- `GL2PS_NO_SORT` The primitives are not sorted, and are output in `stream` in the order they appear in the feedback buffer.
- `GL2PS_SIMPLE_SORT` The primitives are sorted according to their barycenter. This can be sufficient for simple scenes.
- `GL2PS_BSP_SORT` The primitives are inserted in a BSP tree. The tree is then traversed back to front in a painter-like algorithm. This should be used for complex three-dimensional scenes, but keep in mind that the BSP tree algorithm is quite memory hungry...

`options` Sets global plot options, chosen among (multiple options can be combined with the bitwise inclusive OR symbol `|`):

- `GL2PS_NONE` No option.
- `GL2PS_DRAW_BACKGROUND` The background frame is drawn.
- `GL2PS_SIMPLE_LINE_OFFSET` A small offset is added in the z-buffer to all the lines in the plot. This is a simplified version of the `GL2PS_POLYGON_OFFSET_FILL` functionality (cf. section 2.3), putting all the lines of the rendered image slightly in front of their actual position. This thus performs a simple anti-aliasing solution, e.g. for finite element like meshes.
- `GL2PS_SILENT` All the messages written by GL2PS on the error stream are suppressed.
- `GL2PS_BEST_ROOT` The construction of the BSP tree is optimized by choosing the root primitives leading to the minimum number of splits.
- `GL2PS_NO_TEXT` All the text strings are suppressed from output. This is useful to produce the image part of a LaTeX output.
- `GL2PS_LANDSCAPE` Landscape orientation instead of portrait.
- `GL2PS_NO_PS3_SHADING` No use is made of the `shfill` PostScript level 3 operator (which can lead to problems when converting PostScript files to PDF files; see also section 2.5).
- `GL2PS_OCCLUSION_CULL` All the hidden polygons are removed from the output, thus substantially reducing the size of the output file.

`colormode` Specifies the color mode: `GL_RGBA` or `GL_COLOR_INDEX`.

`colorsize` Specifies the size of the colormap if `colormode` is `GL_COLOR_INDEX`.

`colortable` Contains the colormap if `colormode` is `GL_COLOR_INDEX`. This colormap must contain `colorsize` elements of type `GL2PSrgba`.

`buffersize` Specifies the size of the feedback buffer.

`stream` Specifies the stream to which data is printed.

`filename` Specifies a name for the stream to which data is printed.

2.2 gl2psText

2.2.1 Specification

```
void gl2psText( const char *string, const char *fontname,
                GLint fontsize )
```

2.2.2 Description

`gl2psText` permits to include text in the PostScript or LaTeX output in a very simple way. The text is inserted at the current raster position (set by one of the `glRasterPos` OpenGL commands). Beware that text will be sorted according to the position of the leftmost element of the string only. The parameters are:

`string` Specifies the text string to print.

`fontname` Specifies the name of a valid PostScript font (for example `"Times"` or `"HelveticaBoldItalic"`). This parameter has no influence for LaTeX output.

`fontsize` Specifies the size of the font. This parameter has no influence for LaTeX output.

2.3 gl2psEnable and gl2psDisable

2.3.1 Specification

```
void gl2psEnable( GLint mode )
```

```
void gl2psDisable( GLint mode )
```

2.3.2 Description

`gl2psEnable` and `gl2psDisable` delimit OpenGL commands to which a local `mode` is applied. These modes are:

`GL2PS_POLYGON_OFFSET_FILL` Tries to emulate the `GL_POLYGON_OFFSET_FILL` functionality. The value of the offset is taken as the current value of the corresponding OpenGL offset (set with `glPolygonOffset`). Not fully functional yet.

`GL2PS_POLYGON_BOUNDARY` Not implemented yet.

`GL2PS_LINE_STIPPLE` Tries to emulate the `GL_LINE_STIPPLE` functionality.

2.4 gl2psPointSize and gl2psLineWidth

2.4.1 Specification

```
void gl2psPointSize( GLfloat value )
```

```
void gl2psLineWidth( GLfloat value )
```

2.4.2 Description

`gl2psPointSize` and `gl2psLineSize` emulate the standard `glPointSize` and the `glLineWidth` functions. They are necessary since the point sizes and line widths are not saved in the OpenGL feedback buffer.

2.5 gl2psNumShadeColors

2.5.1 Specification

```
void gl2psNumShadeColors( GLint nr, GLint ng, GLint nb )
```

2.5.2 Description

`gl2psNumShadeColors` controls the number flat-shaded (sub-)triangles used to approximate a smooth-shaded triangle when the `shfill` operator is not supported by the system, or when the `GL2PS_NO_PS3_SHADING` option is set. The arguments `nr`, `ng` and `nb` specify the number of values used for interpolating the full range of red, green and blue color components; that is, a triangle is recursively subdivided until the color difference between two of its vertices is smaller that 1/`nr` for the red component, 1/`ng` for the green component and 1/`nb` for the blue component. The last call before `gl2psEndPage` is the one taken into account.

3 Example

Here is a typical calling sequence to produce BSP sorted PostScript output in the file `"MyFile"`, with all the lines slightly shifted front in the z-buffer and all invisible primitives removed to reduce the size of the output file. The `draw()` function contains all the OpenGL commands.

```
FILE *fp = fopen("MyFile", "w");
int buffsize = 0, state = GL2PS_OVERFLOW;

while( state == GL2PS_OVERFLOW ){
  buffsize += 1024*1024;
  gl2psBeginPage ( "MyTitle", "MySoftware",
                   GL2PS_EPS, GL2PS_BSP_SORT,
                   GL2PS_SIMPLE_LINE_OFFSET | GL2PS_SILENT |
                   GL2PS_OCCLUSION_CULL | GL2PS_BEST_ROOT,
                   GL_RGBA, 0, NULL, buffsize, fp, NULL );
  draw();
  state = gl2psEndPage();
}

fclose(fp);
```

To output the text "MyText" at the current raster position, the draw() function should contain something like:

```
gl2psText("MyText", "Courier", 12);
```

If you plan to convert the PostScript file to a PDF file you may need to disable the use of the Level 3 PostScript shfill operator, i.e. add GL2PS_NO_PS3_SHADING to the list of options. Note that you can also edit the output file a posteriori (just set /tryPS3shading to false in the PostScript file header).

A complete example (gl2psTest.c) is included in the distribution.

4 Limitations

GL2PS works by capturing the contents of the OpenGL feedback buffer[2]. As such, all the OpenGL operations applied in the pipeline after the creation of the feedback buffer will be ignored or have to be duplicated by GL2PS (e.g. font rendering, polygon offset or line stippling: see sections 2.2, 2.3 and 2.4).

Other limitations include:

- Rendering large and/or complicated scenes is slow and/or can lead to large output files. This is normal: vector-based images are not destined to replace bitmaps. They just offer an alternative when high quality (especially for 2D and small 3D plots) and ease of manipulation (how do you change the scale, the labels or the colors in a bitmap picture long after the picture was produced, and without altering its quality?) are important.
- Transparency is not supported yet. See the TODO file for some explanations why this may or may not be possible.
- GL2PS does not support textures, volumic rendering, fog effects, etc. One could imagine producing mixed mode vector/bitmap output, but this has not been implemented yet.

5 Contributors

Michael Sweet (mike@easysw.com) for the original implementation of the feedback buffer parser; Bruce Naylor (naylor@comp-graphics.com) for BSP tree and occlusion culling hints; Marc Umé (marc.ume@digitalgraphics.be) for the original list code; Jean-François Remacle (remacle@scorec.rpi.edu) for plane equation fixes; Bart Kaptein (B.L.Kaptein@lumc.nl) for memory leak fixes; Quy Nguyen-Dai (quy@vnilux.com) for output file size optimization; Sam Buss (sbuss@ucsd.edu) for the shfill-based smooth shaded triangle code;

[2] Check the documentation of glRenderMode(GL_FEEDBACK) in your favorite OpenGL reference. See also section 6 for links to other libraries using the same principle.

Shane Hill (Shane.Hill@dsto.defence.gov.au) for the landscape option implementation; Romain Boman (r_boman@yahoo.fr) for the Windows dll generation; Rouben Rostamian (rostamian@umbc.edu) for various bug fixes; Diego Santa Cruz (Diego.SantaCruz@epfl.ch) for the new optimized shaded triangle code and the shfill management; Shahzad Muzaffar (Shahzad.Muzaffar@cern.ch) and Lassi Tuura (lassi.tuura@cern.ch) for the new occlusion culling code and the improvement of GL2PS_BEST_ROOT.

6 Links

Projects similar to GL2PS include: Michael Sweet's GLP library (http://www.easysw.com/~mike/opengl/index.html); Mark J. Kilgard's rendereps (http://www.opengl.org/developers/code/mjktips/Feedback.html); the GLpr library from CEI international (http://www.ceintl.com/; this product does not seem to be available anymore).

Toby White (tow@sdf.lonestar.org) maintains a Python wrapper for GL2PS, available at http://www-theor.ch.cam.ac.uk/people/tow/software.html.

7 Versions

0.1 (Feb 12, 2000) First distributed version.

0.2 (Feb 20, 2000) Added GL2PS_POLYGON_BOUNDARY and GL2PS_BEST_ROOT. Changed arguments of gl2psBeginPage and gl2psText. Corrected some memory allocation stuff. First version of this user's guide.

0.21 (Mar 16, 2000) Initialization fixes.

0.3 (Jul 29, 2000) Code cleaning. Added GL2PS_LINE_STIPPLE.

0.31 (Aug 14, 2000) Better handling of erroneous primitives.

0.32 (May 23, 2001) Fixed memory leaks.

0.4 (Jun 12, 2001) Added gl2psPointSize and gl2psLineWidth. Some code cleaning to allow easier generation of vector file formats other than postscript.

0.41 (Aug 6, 2001) Fixed string allocation (1 char too short). Set smaller default line width.

0.42 (Oct 8, 2001) Optimization of output file size. PostScript header cleaning. Better line width computation.

0.5 (Nov 19, 2001) New format and filename arguments for gl2psBeginPage. Better PostScript handling of smooth shaded primitives. Fix handling of zero-length strings. New options for LaTeX output. Changed (again) the line width computation.

0.51 (Jan 22, 2002) Fixed erroneous drawing of text primitives lying outside the viewport.

0.52 (Feb 14, 2002) New `GL2PS_LANDSCAPE` option.

0.53 (Mar 11, 2002) New `GL2PSDLL` compilation flag to allow the generation of a Windows dll.

0.6 (Jun 4, 2002) Fixed some incoherences in string allocation; fixed sorting of text objects; removed (non functional) occlusion culling code; fixed handling of color and line width attributes when gl2ps was called multiple times inside the same program.

0.61 (Jun 21, 2002) Fixed the fix for the sorting of text objects; introduced tolerance for floating point comparisons.

0.62 (Sep 6, 2002) New `GL2PS_EPS` option to produce Encapsulated PostScript files; optimized drawing of shaded primitives; new `GL2PS_NO_PS3_SHADING` option and `gl2psNumShadeColors` function to control the use of the PostScript level 3 `shfill` operator (usually not well handled when converting to PDF).

0.63 (Nov 12, 2002) Changed `GLvoid` to `void` to accomodate some SUN compilers; made subdivision parameters modifiable a posteriori in the output file; revised documentation.

0.7 (Dec 11, 2002) Occlusion culling (`GL2PS_OCCLUSION_CULL`) is (finally!) working thanks to the great work of Shahzad Muzaffar; enhanced `GL2PS_BEST_ROOT`.

0.71 (Dec 13, 2002) Removed C++ style comments inadvertently left in the code; added example program `gl2psTest.c` to the distribution.

0.72 (Jan 21, 2003) Fixed crash in occlusion culling code; enhanced documentation.